# 采用混合搜索策略的阿奎拉优化算法 *

付小朋 [1]，王　勇 [1,2†]，冯爱武 [1]

(1. 广西民族大学人工智能学院，南宁 530006; 2. 广西混杂计算与集成电路设计分析重点实验室，南宁 530006)

**摘　要：**针对阿奎拉优化算法(AO)存在的不足，提出一种采用混合搜索策略的阿奎拉优化算法(HAO)。首先，利用动态调整函数平衡算法的全局探索与局部开发；其次，利用混沌自适应权重来增强算法的全局搜索能力、加快算法的收敛速度；最后，设计新的个体变异概率系数，采用改进型差分变异策略，利用适应度值较优个体引领群体中其他个体开展搜索活动,保持了种群的多样性,增强了算法跳出局部最优能力。通过 8 个基准测试函数和 10 个 CEC2019 测试函数，以及 1 个工程应用问题的数值实验仿真，实验结果表明，算法的全局收敛速度和优化精度均得到了明显地改善，跳出局部最优的能力得到了增强。

**关键词：**阿奎拉优化算法；动态调整；混沌自适应权重；改进型差分变异

**中图分类号：**TP18　　doi: 10.19734/j.issn.1001-3695.2022.04.0128

# Aquila optimization algorithm using hybrid search strategies

Fu Xiaopeng[1], Wang Yong[1, 2†], Feng Aiwu[1]

(1. *College of Artificial Intelligence*, *Guangxi Minzu University*, *Nanning* 530006, *China*; 2. *Guangxi Key Laboratory of Hybrid Computation & IC Design Analysis Nanning* 530006, *China*)

**Abstract:** Aiming at the shortcomings of aquila optimization algorithm(AO) , this paper proposed an aquila optimization algorithm using hybrid search strategy. Firstly, the algorithm introduced dynamic adjustment function to balance global exploration and local exploitation; Secondly, it introduced chaotic adaptive weights to enhance the global search capability of the algorithm and accelerate the convergence speed of the algorithm; Thirdly, it introduced a new individual mutation probability coefficient and an improved differential mutation strategy, and used individuals with better fitness values to lead other individuals in the population to carry out search activities, which maintained the diversity of the population and enhanced the ability of the algorithm to jump out of the local optimum. Through the numerical experiment simulation of 8 benchmark test functions, 10 CEC2019 test functions and 1 engineering application problem, The experimental results show that the algorithm has a significant improvement in global convergence speed and optimization accuracy, and has a better ability to jump out of the local optimum.

**Key words:** aquila optimization algorithm; dynamic adjustment; chaotic adaptive weights; improved differential mutation



## 0　引言

在过去的几十年里，元启发式算法已经被成功地应用于解决科学、工程等领域的优化问题。自 Holland 提出遗传算法(GA)[1]以来，国内研究者已经提出了包括差分进化算法(DE)[2]、粒子群算法(PSO)[3]、鲸鱼优化算法(WOA)[4]、引力搜索算法(GSA)[5]、正余弦算法(SCA)[6]、教学优化算法(TLBO)[7]、社会进化算法(SEA)[8]等众多元启发式算法。目前，元启发式算法已逐渐在机器人路基规划[9]、图像分割[10]、集群车辆路径问题[11]等领域得到了广泛地应用。

阿奎拉优化算法(Aquila optimizer, AO)[12]是 Abualigah 等人于 2021 年提出的一种新的元启发式算法。该算法启发于“鹰科”飞行猛禽阿奎拉的狩猎行为，具有原理简单、易于编程实现等特点。然而,AO 也明显存在收敛速度慢、易陷入局部最优等不足。针对 AO 的不足,文献[13]采用将 AO 与哈里斯鹰算法相混合的改进策略,以增强算法的局部开发能力,文献[13]还通过引入非线性逃逸能量参数和随机对立学习策略来增强算法跳出局部最优能力；文献[14]通过引入混沌映射来提高算法搜索的随机性，把饥饿游戏搜索算法与 AO 相混合，以增强算法的探索能力；文献[15]则采用对立学习策略和小波变异策略来提升算法的优化精度；文献[16]则将 AO 与算术优化算法相结合，以提高算法的搜索效率。以上文献[13~16]提出的改进版本 AO，相较之于标准 AO，其在优化精度方面有所改善，,然而这些改进版本 AO 依然存在全局搜索能力不强、规避陷入局部最优的能力不强、早熟收敛等问题，仍有待进一步完善。针对这一问题，本文提出一种新的采用混合搜索策略的阿奎拉优化算法(Aquila optimization algorithm using hybrid search strategies，HAO)，首先引入动态调整策略平衡算法的勘探与开发，并利用混沌自适应权重提升算法的探索与开发能力，其次在算法陷入局部最优时，引入改进型差分变异策略对非最优个体进行变异，提升算法多样性，避免算法陷入局部最优。通过数值实例仿真验证了本文采用的改进策略的有效性和可行性。

## 1　AO 算法简介

AO 算法的原理源于阿奎拉狩猎行为之间的切换，阿奎拉有四种狩猎行为：一是垂直俯冲的高空飞行模式，广泛的确定狩猎区域和位置；二是短滑翔攻击的等高线飞行，进一步探索猎物所在区域；三是缓慢下降攻击的低空飞行，在选定的目标区域接近猎物并进行攻击；四是降落地面，依靠行走发起精确攻击。相应地，在 AO 算法的数学模型中，阿奎拉的前两种狩猎行为表示为全局探索，分别表示为扩展探索

**收稿日期：**2022-04-07；**修回日期：**2022-05-18　**基金项目：**国家自然科学基金项目(61662005)；广西自然科学基金项目(2021JJA170094)

**作者简介：**付小朋(1996-)，男，重庆人，硕士研究生，主要研究方向为计算智能；王勇(1963-)，男(通信作者)，广西南宁人，教授，硕导，博士，主要研究方向为计算智能(wangygxnn@sina.com)；冯爱武(1996-)，男，山西临汾人，硕士研究生，主要研究方向为计算智能.

和收缩探索；后两种狩猎行为表示为局部开发，分别表示为扩展开发和收缩开发。其中，全局探索基于条件 $t\le 2/3T$ 执行($t$ 为算法当前迭代次数，$T$ 为最大迭代次数)，否则执行局部开发。

### 1.1　全局探索

a) 扩展探索。在这个阶段阿奎拉通过高空飞行能广泛的探索目标所在区域。在全局探索阶段基于条件 $rand\le 0.5$($rand$ 为 0 到 1 之间的随机数)执行扩展探索，否则执行收缩探索。扩展探索的数学模型如式(1)所示。

$$X(t+1)=X_{best}(t)\times(1-t/T)+(X_{M}(t)-rand*X_{best}(t)) \quad (1)$$

$$X_M(t)=\frac{1}{N}\sum_{i=1}^{N}X_i(t),\quad \forall\ j=1,2,\ldots,Dim \quad (2)$$

其中：$X_{best}(t)$表示第 t 次迭代最佳解，$1-t/T$ 表示通过迭代次数来调整探索范围，$X_i(t)$表示第 $t$ 次迭代第 $i$ 个体，$X_M(t)$为第 $t$ 次迭代的均值，$Dim$ 表示问题维度大小，$N$ 为种群大小。

b) 收缩探索。在这个阶段通过等高线飞行进一步探索目标区域。其收缩探索的数学模型如式(3)所示。

$$X(t+1)=X_{best}(t)\times levy(Dim)+X_R(t)+rand*(y-x) \quad (3)$$

$$levy(Dim)=(s\times u\times\sigma)/|v|^{1/\beta} \quad (4)$$

$$\sigma=\left(\frac{\Gamma(1+\beta)\times\sin(\pi\beta/2)}{\Gamma(1+\beta)\times\beta\times 2^{(\beta-1)/2}}\right) \quad (5)$$

$$y=r\times\cos(\theta),\ x=r\times\sin(\theta) \quad (6)$$

其中:$levy(Dim)$表示莱维飞行的分布函数，$X_R(t)$表示当前种群的随机个体,固定常量 $s=0.01$，$u$ 和 $v$ 均为[0,1]中的随机数，$\beta=1.5$。$y$ 和 $x$ 表示搜索过程中的螺旋形状，$r=r_1+U\times D_1$，$\theta=3\pi/2-\omega\times D_1$,$r_1$ 为 1 到 20 的固定值,表示固定的搜索周期数。$U=0.00565$,$D_1$ 为一个 1 到搜索空间维度($Dim$)的整数值，$\omega=0.005$。

### 1.2　局部开发

a) 扩展开发。在这个阶段，阿奎拉通过低空飞行开始对目标进行初步攻击。在局部开发阶段基于条件 $rand\le 0.5$ 执行扩展开发，否则执行收缩开发。其扩展开发的数学模型如式(7)所示。

$$X(t+1)=(X_{best}(t)-X_M(t))\times\alpha-rand+((UB-LB)\times rand+LB)\times\delta \quad (7)$$

其中：$UB$,$LB$ 表示给定问题的上界和下界，$\alpha$ 和 $\delta$ 为开发调整参数(AO 中均为 0.1)。

b)收缩开发。在这个阶段，阿奎拉降落地面对目标发起精确攻击。其收索开发的数学模式如式(8)所示。

$$X(t+1)=QF(t)\times X_{best}(t)-(G_1\times X(t)\times rand)-G_2\times levy(D)+rand\times G_1 \quad (8)$$

$$QF(t)=t^{\frac{2\times rand-1}{(1-T)^2}},G_1=2\times rand-1,G_2=2\times(1-\frac{t}{T}) \quad (9)$$

其中:$QF$ 用于平衡搜索策略的质量函数，$G_1$ 表示追踪潜逃猎物的各种活动，$G_2$ 表示追踪猎物时的飞行斜率。

## 2　HAO 算法

在标准 AO 算法中，首先，全局探索与局部开发之间的切换基于条件 $t\le(2/3)T$ 这一固定概率转换，容易导致算法探索与开发之间的不平衡，探索与开发之间缺乏信息交流。在算法搜索过程中在前期通过全局探索广泛确定目标所在区域，其仅仅只通过全局探索步骤搜索，未与局部开发步骤进行信息交流来缩小探索区域，容易导致算法在前期确定的目标区域过大；在算法后期，又通过在前期确定的大范围区域进行精确的局部开发搜索，其容易导致搜索效率的下降，算法不能快速确定最优目标，算法收敛速度下降，收敛精度不高。其次，在全局探索与局部开发阶段中，种群中个体的更新都围绕种群最优个体进行，其容易导致算法的寻优性能对当前种群中最优个体的位置过度依赖，如果种群最优个体过早陷入局部最优的位置，其往往会对其他非最优个体进行引导聚集，过早的造成算法多样性下降，导致算法出现早熟现象，算法陷入局部最优。基于以上分析，本文提出一种采用混合搜索策略的阿奎拉优化算法，以提高 AO 算法的优化性能。

### 2.1　动态调整

固定的探索与开发概率容易导致搜索之间的不平衡，进而降低算法的搜索效率。AO 算法中不难看出，算法在执行探索步骤与开发步骤之间缺乏信息交流，导致探索与开发各自为政，探索与开发之间存在严重不平衡。为了克服这一缺陷，本文通过一种震荡变化的函数作为切换探索与开发之间的动态调整概率系数。动态调整的函数如式(10)所示，其中动态调整函数变化曲线,如图 1 所示。

$$r(t)=(exp(-5t/T)\times\cos(5t)+l)/2 \quad (10)$$

其中：$l=1$,当 $r(t)<0.5$ 时，其值变换为 $1-r(t)$。在改进算法中通过 $rand\le r(t)$动态切换全局探索步骤与局部开发步骤，基于一般元启发式算法普遍采用“算法在搜索前期侧重进行全局勘探、在后期侧重开展局部开发”搜索策略原则，HAO 算法也遵循这一原则。从图 1 可以看出，$r(t)$的值在迭代前期表现为远离数值 0.5 波动变化，随着迭代次数的增加，$r(t)$值的波动幅度逐渐靠近数值 0.5，最后 $r(t)$在数值 0.5 附近波动变化，算法在前期 $r(t)$的值大于 $rand$ 的概率大，在前期执行全局探索的概率较大，随着迭代次数的增加，$r(t)$波动数值减小，局部开发的概率逐渐增加，最后，全局探索与局部开发的执行概率为一种动态平衡。通过 $rand\le r(t)$这一切换原则，算法在前期既能以较大概率执行全局探索，进行目标收敛区域的确定，又能有较小概率执行局部开发，以小概率进行开发，可进一步精确确定目标收敛区域，于算法前期确定的收敛范围将更加接近目标区域,收敛范围也更加有效;到了算法后期，由于是在更靠近目标的附近实施探索与开发行为，其找到目标的概率也随之增大，算法的收敛速度和优化精度也随之得到了提高。本文算法(HAO)由于利用切换式 $r(t)$来调整算法的全局探索与局部开发,使算法在全局探索与局部开发之间可动态相互交流与交叉切换，从而可重新调整目标收敛区域，避免由于盲目搜索而导致算法收敛速度和收敛精度之不高。

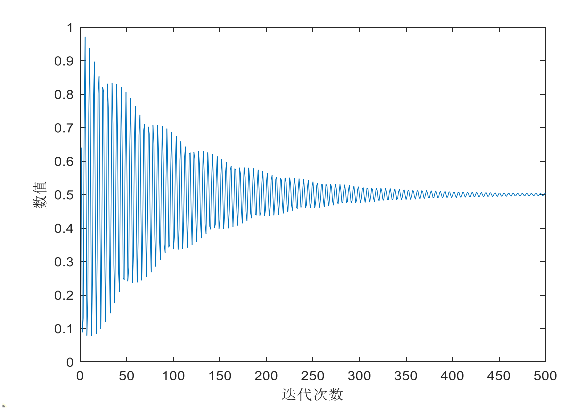



图 1　动态调整函数变化曲线

Fig. 1　Dynamic adjustment function change curve

### 2.2　混沌自适应权重

较大的惯性权重有利于算法的探索行为，较小的惯性权重有利于算法的开发行为。为此，本文设计一种非线性递减的惯性权重，并引入 *logistic* 混沌系数的混沌自适应权重。混沌自适应权重的公式如式(11)所示，混沌自适应权重的变化曲线如图 2 所示。

$$w(t)=z(t)\times w_1-(w_1-w_2)\times(2t/T-(t/T)^2) \quad (11)$$

$$z(t)=a\times z(t-1)\times(1-z(t-1)) \quad (12)$$

其中：$w_1=0.9$，$w_2=0.4$,$a=3$,$z(t)$表示 *logistic* 混沌系数。从图 2

可以看出混沌自适应权重的变化曲线整体上是非线性递减，这样的策略既使得算法在前期有着高效的探索效率，加快算法目标收敛区域的确定，在算法后期又能提升算法开发效率，快速确定目标位置，提升算法收敛速度。其中混沌系数具有随机性的特点，从图 2 可以看出，混沌系数的引入使得自适应权重的变化具有一定的灵活性，自适应权重按波动性的非线性的规律变化。自适应权重通过对当前最优个体进行波动性扰动，可以有效提升当前最优个体的灵活性，使当前最优个体对种群其他个体的引导更加灵活，有助于种群多样性的提升，避免算法陷入局部最优，从而提升算法收敛精度。其中对式(1)(3)(7)(8)引入混沌自适应权重之后，分别如式(13)(14)(15)(16)。

$$X(t+1)=w(t)\times X_{best}(t)\times(1-t/T)+(X_M(t)-rand*X_{best}(t)) \quad (13)$$

$$X(t+1)=w(t)\times QF(t)\times X_{best}(t)-(G_1\times X(t)\times rand)-G_2\times levy(D)+rand\times G_1 \quad (14)$$

$$X(t+1)=w(t)\times(X_{best}(t)-X_M(t))\times\alpha-rand+((UB-LB)\times rand+LB)\times\delta \quad (15)$$

$$X(t+1)=w(t)\times QF(t)\times X_{best}(t)-(G_1\times X(t)\times rand)-G_2\times levy(D)+rand\times G_1 \quad (16)$$

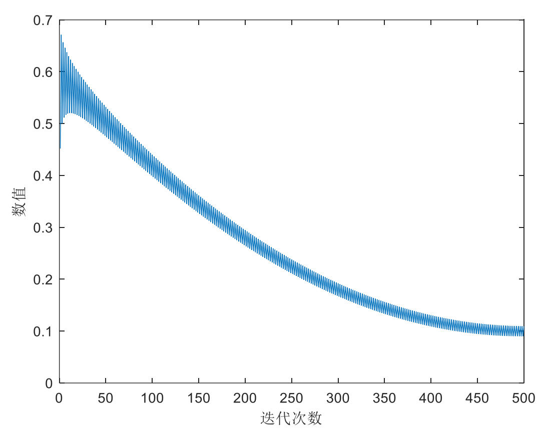



图 2　混沌自适应权重变化曲线

Fig. 2　Chaotic adaptive weight curve

## 2.3　改进型差分变异策略

从 AO 的式(1)(3)(7)和(8)不难看出，其具有一个共同点，都是利用当前最优个体进行引导的搜索，基于前面的分析，其容易导致种群多样性下降，算法陷入局部最优。因此，提升种群多样性是算法避免陷入局部最优的关键所在。本文利用改进型差分变异策略，对种群中非最优个体进行变异操作，以此增加种群的多样性，增强算法跳出局部最优的能力。

a) 变异概率系数的设计。本文主要对非最优个体进行变异操作，采用传统的固定变异概率系数具有一定的随机性与盲目性。因此，本文设计一种新的变异概率系数，其具有明确的指向性，使变异向有助于算法收敛的方向进行，进而加快算法的收敛。其变异概率系数如式(17)所示。

$$p(i)=\begin{cases}0, & fit_i=fit_{min}\\ \dfrac{0.3(fit_i-fit_{min})}{fit_{max}-fit_{min}}, & fit_i<(fit_{max}+fit_{min})/2\\ 0.3, & fit_i\ge(fit_{max}+fit_{min})/2\end{cases} \quad (17)$$

其中：$p(i)$表示第 $i$ 个体的变异概率，$fit_i$ 表示第 $i$ 个体的适应度值，$fit_{min}$ 表示群体中最小适应度值，$fit_{max}$ 表示群体的最大适应度值，当个体的变异概率 $rand< p(i)$时则执行变异策略。式(17)表示：对最优个体的变异概率为 0，对非最优个体来说，其离最优个体越远，变异的概率就越大；反之，则越小。因此，该变异概率系数是具有指向性的，可使非最优个体的变异朝算法收敛的方向进行。

b) 变异操作。本文采用改进型差分变异策略进行变异操作，以提升算法的多样性。差分变异方法主要有：DE/rand/1、DE/rand/2 、DE/best/1 、DE/best/2 、DE/current-to-rand/1 、DE/current-to-best/1 等。本文选取 DE/best/2 并加以改进，然后将其用于 AO 的变异操作。DE/best/2 变异公式如式(18)所示。

$$X(t+1)=X_{best}(t)+F\times(X_{r1}(t)-X_{r2}(t))+F\times(X_{r3}(t)-X_{r4}(t)) \quad (18)$$

其中：$X_{best}$ 为群体当前最优个体，$X_{r1}$、$X_{r2}$、$X_{r3}$、$X_{r4}$ 为种群中的随机个体，$F$ 为比例因子，为 0 到 1 之间的随机数。本文对式(18)作如下改进(见式(19):

$$Xnew(t+1)=\frac{X_\alpha(t)+X_\beta(t)+X_\delta(t)}{3}+F\times\mathrm{sgn}(m1)\times(X_{r1}(t)-X_{r2}(t))+F\times\mathrm{sgn}(m2)\times(X_{r3}(t)-X_{r4}(t)) \quad (19)$$

$$m1=fit(X_{r1}(t))-fit(X_{r2}(t)),\ m2=fit(X_{r3}(t))-fit(X_{r4}(t)) \quad (20)$$

其中：$fit(X)$表示 $X$ 的适应度值，$sgn(\bullet)$为符号函数。式(19)首先将 $X_{best}$ 改为种群中个体适应度前三个体的均值。通过适应度前三个体进行引导，个体的更新将不会盲目的趋向最优个体附近，且个体的分布在搜索空间将更加灵活，从而整个种群的多样性将会得到改善，种群多样性增加有利于算法跳出局部最优，从而提升算法的收敛精度；其次，在扰动项上增加符号函数，利用随机个体的适应度信息来增强扰动项的离散性，个体的更新将更加灵活，种群多样性进一步得到改善。通过式(13)变异后的 $Xnew$ 与当前个体进行适应度值对比，如果变异后适应度值更优，则保留变异个体。

## 2.4　HAO 算法流程

综上所述，HAO 算法执行的流程如图 3 所示。

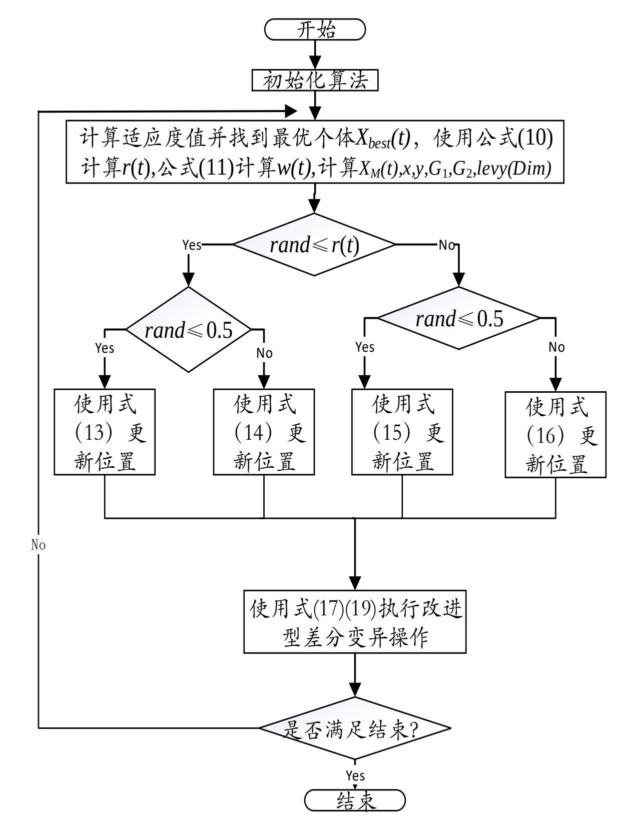



图 3　HAO 算法流程图

Fig. 3　HAO algorithm flowchart

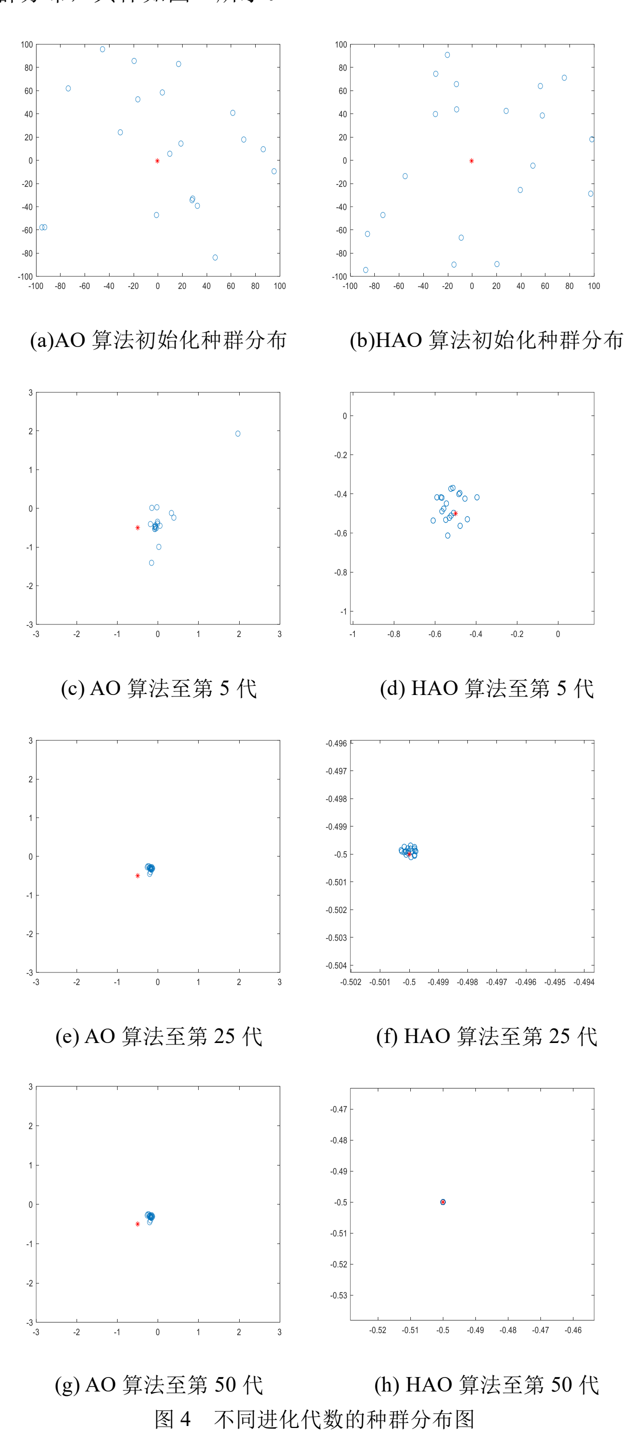
## 2.5　HAO 算法时间复杂度分析

假设算法迭代次数为 $T$，种群规模为 $N$，搜索空间维度为 $D$，标准 AO 算法的时间复杂度为 $o(N\times(T\times D+1))$。HAO 算法中，使用动态调整的计算时间复杂度为 $o(T)$，使用混沌自适应权重的计算时间复杂度为 $o(T)$，在改进差分变异中，变异概率系数的计算时间复杂度为$(T\times N)$，变异操作的计算时间复杂度为$(N\times T\times D)$。若不考虑计算时间复杂度之低次

项，则 HAO 的计算时间复杂度为 $o(N\times(T\times D+1))$，与标准 AO 的计算时间复杂度一致。

### 2.6 种群分布分析

设搜索空间为 2 维，种群规模为 20。本文利用 Step 函数来描绘和刻画 HAO 算法和 AO 算法在不同进化代数的种群分布，具体如图 4 所示。



(a)AO 算法初始化种群分布　(b)HAO 算法初始化种群分布

(c) AO 算法至第 5 代　(d) HAO 算法至第 5 代

(e) AO 算法至第 25 代　(f) HAO 算法至第 25 代

(g) AO 算法至第 50 代　(h) HAO 算法至第 50 代

图 4　不同进化代数的种群分布图

Fig. 4　Population distribution map of different evolutionary generations

注：图 4 中的红色小点为 Step 函数的理论最优位置、蓝色小圈为种群中不同个体的位置。从图 4 可以看出：进化至第 5 代，AO 算法种群中大多数个体聚集分布在与红色小点(理论最优位置)还有一段距离的位置，但没有个体靠近红色小点；HAO 算法种群中大多数个体都已经分布在红色小点(理论最优位置)的附近周围，并且有部分个体已经非常接近红色小点了。进化至第 25 代，AO 算法种群大多数个体仍然聚集在与红色小点(理论最优位置)还有一段距离的位置，也没有个体到达红色小点位置；HAO 算法种群中大多数个体都已经非常接近红色小点，并且有部分个体已经到达红色小点位置，算法已经找到理论最优位置了。进化至第 50 代，AO 算法种群的分布情况与至第 25 代种群的分布情况基本一致，算法已陷入局部最优；HAO 算法种群中大多数个体都已经到达红色小点位置(理论最优位置)。综合以上分析，从另一个侧面说明 HAO 算法具有较强的全局搜索能力和较快的全局收敛速度，种群的多样性得到有效保持，规避陷入局部最优的能力较强。

## 3 数值实验仿真分析

### 3.1 实验环境配置与基准测试函数

本次实验仿真的环境配置为：64 位 Win10 操作系统,Intel(R)Core(TM)i7-1065G7 的 CPU，主频 1.30GHz，内存 16G,仿真软件为 MatlabR2019b。

为全面分析改进算法的性能，本文将 HAO 与标准 AO 算法、海鸥优化算法 SOA[17]、算术优化算法(AOA)[18]，哈里斯鹰优化算法(HHO)[19]，文献[15]的 IAO 算法、文献[16]的 AOAAO 算法进行数值仿真实验对比。选取 8 个基准测试函数进行数值实验测试，测试函数如表 1 所示。

表 1　测试函数

Tab. 1　Test functions

| 函数编号 | 函数名 | 类型 | 维度 | 范围 | 理论值 |
|---|---|---|---|---|---|
| $f_1$ | Sphere | 单峰 | 30 | [-100,100] | 0 |
| $f_2$ | Schwefel2.22 | 单峰 | 30 | [-10,10] | 0 |
| $f_3$ | Schwefel1.2 | 单峰 | 30 | [-100,100] | 0 |
| $f_4$ | Schwefel2.21 | 单峰 | 30 | [-100,100] | 0 |
| $f_5$ | Step | 单峰 | 30 | [-100,100] | 0 |
| $f_6$ | Griewank | 多峰 | 30 | [-600,600] | 0 |
| $f_7$ | Penalized 2 | 多峰 | 30 | [-50,50] | 0 |
| $f_8$ | Penalized 1 | 多峰 | 30 | [-50,50] | 0 |

### 3.2 数值实验分析与收敛性分析

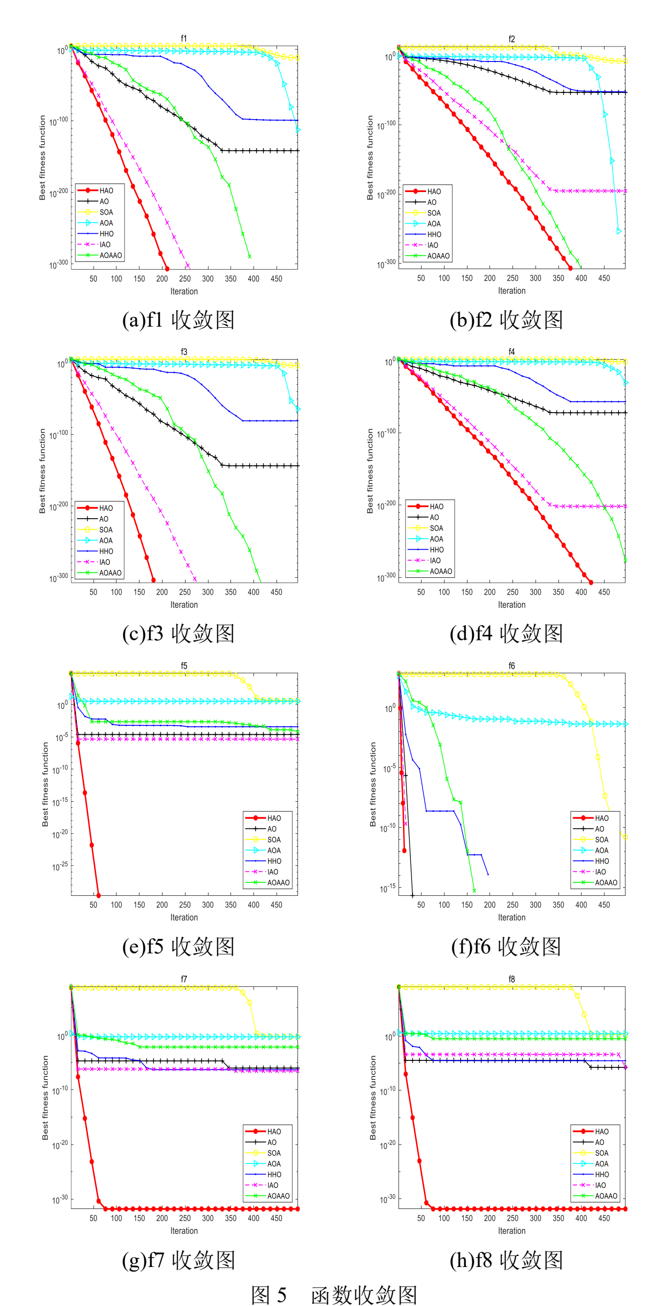
为了实验测试的公平性，实验中，所有对比算法均在同一条件下运行。其中：种群规模为 30，最大迭代次数为 500。在本文测试数据中，为避免随机性对测试结果的干扰，每一算法对每一测试函数都独立运行 30 次，并记录每次运行测试的平均值和标准差。其中：在平均值指标上精度越高，表明算法的寻优能力越强；在标准差指标上精度越高，表明算法整体寻优稳定性越优。表 2 为所有对比算法在 8 个基准函数测试数据的对比。为比较算法的收敛速度，将 HAO 与其他 6 种对比算法作收敛速度测试(同一条件下)，得到的函数收敛图如图 5 所示。

基于表 2 实验数据作对比分析：从平均值指标上看，HAO 在 $f_1$- $f_6$ 均达到理论最优值，在未达到理论最优的 $f_7$ 和 $f_8$ 上，HAO 的精度均高于其他 6 种对比算法。值得强调的是 $f_7$ 为高维多峰函数，寻找其最优值比较难，然而，HAO 相较于 AO 在平均值寻优精度上提升了 8 个量级，提升效果较为明显。从表 2 上看，HAO 寻优能力均强于对比算法，其中：AO 和 HHO 仅在 $f_6$ 上找到的平均值达到理论最优；SOA 和 AOA 均未找到理论最优；IAO 在 $f_1$、 $f_3$ 、$f_7$ 找到理论最优，AOAAO 在 $f_1$- $f_3$、 $f_7$ 上找到理论最优值。因此，HAO 的寻优能力在对比算法当中是最强的。从标准差指标上看，HAO 得到的标准差精度均比其他对比算法的好，其中：HAO 求解 $f_1$- $f_6$ 得到的标准差为理论最优，表现均比其他对比算法的好，求解 $f_7$, $f_8$ 得到的标准差也比其他对比算法更接近理论最优值，因而，HAO 的算法稳定性均比其他对比算法的好。HAO 优化精度的提高主要得益于采用了改进型差分变异策略，通过随机个体的差分作用，并利用适应度值较优的多个个体进行引领，有效提升了种群多样性，增强了算法跳出局部最优能力的同时，又可提升算法的优化精度。

表 2　基准函数测试数据

Tab. 2　Benchmark function test data

| 函数 | 指标 | HAO | AO | SOA | AOA | HHO | IAO | AOAAO |
|---|---|---|---|---|---|---|---|---|
| $f_1$ | 平均值 | **0.00E+00** | 8.86E-102 | 4.07E-12 | 8.00E-28 | 6.19E-95 | **0.00E+00** | **0.00E+00** |
| | 标准差 | **0.00E+00** | 3.43E-101 | 7.02E-12 | 3.10E-27 | 2.35E-94 | **0.00E+00** | **0.00E+00** |
| $f_2$ | 平均值 | **0.00E+00** | 2.91E-56 | 2.45E-08 | 0.00E+00 | 3.33E-49 | 2.54E-192 | **0.00E+00** |
| | 标准差 | **0.00E+00** | 1.13E-55 | 2.08E-08 | 0.00E+00 | 9.23E-49 | **0.00E+00** | **0.00E+00** |
| $f_3$ | 平均值 | **0.00E+00** | 2.42E-131 | 2.08E-05 | 3.11E-03 | 1.87E-78 | **0.00E+00** | **0.00E+00** |
| | 标准差 | **0.00E+00** | 9.35E-131 | 2.33E-05 | 5.91E-03 | 7.22E-78 | **0.00E+00** | **0.00E+00** |
| $f_4$ | 平均值 | **0.00E+00** | 6.29E-63 | 8.86E-04 | 3.20E-02 | 4.63E-48 | 5.54E-196 | 1.07E-244 |
| | 标准差 | **0.00E+00** | 2.44E-62 | 7.12E-04 | 1.82E-02 | 1.31E-47 | 0.00E+00 | 0.00E+00 |
| $f_5$ | 平均值 | **0.00E+00** | 7.81E-03 | 3.22E+00 | 3.31E+00 | 6.78E-05 | 8.89E-05 | 2.87E-05 |
| | 标准差 | **0.00E+00** | 6.31E-03 | 2.44E-01 | 3.53E-01 | 6.31E-05 | 8.91E-05 | 2.62E-05 |
| $f_6$ | 平均值 | **0.00E+00** | **0.00E+00** | 1.71E-02 | 2.43E-01 | **0.00E+00** | **0.00E+00** | **0.00E+00** |
| | 标准差 | **0.00E+00** | **0.00E+00** | 2.66E-02 | 1.68E-01 | **0.00E+00** | **0.00E+00** | **0.00E+00** |
| $f_7$ | 平均值 | **5.60E-14** | 4.56E-06 | 3.76E-01 | 5.47E-01 | 9.15E-06 | 1.35E-06 | 3.19E-03 |
| | 标准差 | **2.17E-13** | 4.47E-06 | 2.23E-01 | 4.79E-02 | 9.76E-06 | 1.91E-06 | 6.20E-03 |
| $f_8$ | 平均值 | **2.54E-13** | 2.12E-05 | 2.05E+00 | 2.85E+00 | 9.20E-05 | 4.93E-05 | 2.01E+00 |
| | 标准差 | **7.51E-13** | 4.12E-05 | 2.15E-01 | 8.64E-02 | 1.85E-04 | 5.97E-05 | 1.27E+00 |




(a)f1 收敛图　(b)f2 收敛图

(c)f3 收敛图　(d)f4 收敛图

(e)f5 收敛图　(f)f6 收敛图

(g)f7 收敛图　(h)f8 收敛图

图 5　函数收敛图

Fig. 5　Function convergence diagram

从收敛图 5 上看：HAO 的收敛曲线(针对所有函数)均位于所有对比算法之收敛曲线的最下方位置，因此，在 7 种对比算法当中，HAO 的收敛速度是最快的，本文算法(HAO)的搜索效率在 7 种对比算法当中是最高的。HAO 具有较好收敛性能，这主要是由于本文算法(HAO)采用的动态调整策略，平衡了算法的全局探索与局部开发，利用混沌自适应权重提高了算法的全局搜索效率，从而加快了算法的收敛速度。

### 3.3　不同改进策略分析

为验证不同改进策略对 HAO 的影响，本文将 HAO 和 AO 分别与仅采用动态调整策略的阿奎拉优化算法(HAO-1)、仅采用混沌自适应权重的阿奎拉优化算法(HAO-2)、仅采用改进型差分变异策略的阿奎拉优化算法(HAO-3)进行数值实验对比分析，并记录平均值(M)和标准差(S)。其中：实验条件与 3.2 节保持一致。不同改进策略对比的实验结果见表 3。

从实验结果表 3 中可以看出：三种改进策略对提高算法的寻优能力都起明显的作用，特别是求解 $f_1$－$f_4$ 时的提升程度较为明显；求解 $f_5$、$f_7$、$f_8$ 时，仅采用动态调整策略的 AO 算法(HAO-1)与仅采用混沌自适应权重的 AO 算法(HAO-2)的提升程度不是非常明显，但是仅采用改进型差分变异策略的 AO 算法(HAO-3)求解这些测试函数时，寻优能力提升程度还是比较明显的。融合三种搜索策略的 HAO，其寻优能力明显优于采用单一搜索策略的 AO 算法，HAO 算法的改进策略对算法寻优性能的提升是具有积极作用的。

表 3　不同改进策略对比

Tab. 3　Comparison of different improvement strategies

| 函数 | 指标 | HAO | AO | HAO-1 | HAO-2 | HAO-3 |
|---|---|---|---|---|---|---|
| $f_1$ | M | **0.00E+00** | 8.86E-102 | 3.00E-284 | 0.00E+00 | 2.52E-286 |
| | S | **0.00E+00** | 3.43E-101 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| $f_2$ | M | **0.00E+00** | 2.91E-56 | 4.48E-130 | 9.17E-212 | 5.30E-147 |
| | S | **0.00E+00** | 1.13E-55 | 1.74E-129 | 0.00E+00 | 1.67E-146 |
| $f_3$ | M | **0.00E+00** | 2.42E-131 | 7.89E-254 | 0.00E+00 | 3.43E-287 |
| | S | **0.00E+00** | 9.35E-131 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| $f_4$ | M | **0.00E+00** | 6.29E-63 | 2.47E-127 | 3.01E-209 | 8.51E-147 |
| | S | **0.00E+00** | 2.44E-62 | 9.55E-127 | 0.00E+00 | 3.17E-146 |
| $f_5$ | M | **0.00E+00** | 7.81E-03 | 2.69E-05 | 6.05E-05 | 0.00E+00 |
| | S | **0.00E+00** | 6.31E-03 | 2.12E-05 | 6.33E-05 | 0.00E+00 |
| $f_6$ | M | **0.00E+00** | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| | S | **0.00E+00** | 0.00E+00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| $f_7$ | M | **5.60E-14** | 4.56E-06 | 7.08E-06 | 1.51E-05 | 1.57E-13 |
| | S | **2.17E-13** | 4.47E-06 | 1.37E-05 | 2.21E-05 | 2.83E-13 |
| $f_8$ | M | **2.54E-13** | 2.12E-05 | 1.07E-04 | 1.79E-04 | 1.35E-13 |
| | S | **7.51E-13** | 4.12E-05 | 1.47E-04 | 2.66E-04 | 2.83E-12 |

### 3.4 Wilcoxon 数学统计检验

本文通过 Wilcoxon 数学统计检验的方法，验证 HAO 与 AO、SOA、AOA、HHO、IOA、AOAAO 的显著性差异。其中：表 4 为 Wilcoxon 检验结果。在显著性水平为 5%条件下检验算法之间的显著差异性，其中 $p$ 代表检验的数值，若 $p$ 值小于 5%，则代表算法之间存在显著差异，否则不具有明显差异。$h$ 为检验结果，$h$ 为 1 则代表有明显差异，为 0 则代表无明显差异。*NaN* 表示数据无效，即检验的所有样本数据相同，算法不具有显著差异性。

从表 4 数据上看：$p$ 值小于 5%的占大多数，因此，HAO 与其他 6 种对比算法具有显著性差异，HAO 的寻优能力更强。

表 4　Wilcoxon 检验结果

Tab. 4　Wilcoxon test results

| 函数 | 指标 | AO | SOA | AOA | HHO | IAO | AOAAO |
|---|---|---|---|---|---|---|---|
| $f_1$ | $p$ | 1.21E-12 | 1.21E-12 | 1.21E-12 | 1.21E-12 | *NaN* | *NaN* |
| | $h$ | 1 | 1 | 1 | 1 | 0 | 0 |
| $f_2$ | $p$ | 1.21E-12 | 1.21E-12 | *NaN* | 1.21E-12 | 1.21E-12 | *NaN* |
| | $h$ | 1 | 1 | 0 | 1 | 1 | 0 |
| $f_3$ | $p$ | 1.21E-12 | 1.21E-12 | 1.2E-12 | 1.21E-12 | *NaN* | *NaN* |
| | $h$ | 1 | 1 | 1 | 1 | 0 | 0 |
| $f_4$ | $p$ | 1.21E-12 | 1.21E-12 | 1.21E-12 | 1.21E-12 | 1.21E-12 | 1.21E-12 |
| | $h$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $f_5$ | $p$ | 1.21E-12 | 1.21E-12 | 1.2E-12 | 1.21E-12 | 1.21E-12 | 1.21E-12 |
| | $h$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $f_6$ | $p$ | *NaN* | 1.21E-12 | 1.21E-12 | *NaN* | *NaN* | *NaN* |
| | $h$ | 0 | 1 | 1 | 0 | 0 | 0 |
| $f_7$ | $p$ | 1.92E-12 | 1.72E-12 | 1.72E-12 | 1.72E-12 | 2.41E-12 | 1.72E-12 |
| | $h$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $f_8$ | $p$ | 1.72E-12 | 1.72E-12 | 1.72E-12 | 1.72E-12 | 1.72E-12 | 1.72E-12 |
| | $h$ | 1 | 1 | 1 | 1 | 1 | 1 |

### 3.5 与最新改进元启发式算法对比

为进一步验证 HAO 算法的寻优性能，将 HAO、AO 算法与文献[20]的采用混合策略改进的学生心理优化算法(SPBO)与文献[21]采用多模式飞行的乌鸦搜索算法(MFCSA)进行数值实验测试对比，并记录测试的平均值与标准差。其中，在本次测试中，选取 CEC2019 的 10 个测试函数进行测试，CEC2019 测试函数具有问题规模大，寻优较为复杂的特性，可以有效区别算法寻优能力之间的差异，其中测试条件与 3.2 节保持一致，CEC2019 测试函数的具体特征如表 5 所示。CEC2019 测试的数据如表 6 所示。

基于表 6 的数据可知，HAO 算法在测试函数 CEC01、CEC02、CEC04-CEC09 的平均值和标准差精度均优于 AO、SPBO、MFCSA 算法，在 CEC03 上所有算法的平均值精度是一致的，但是 HAO 标准差精度好过其他几种对比算法，在 CEC10 上，HAO 算法的标准差精度略差于 SPBO 和 MFCSA，但 HAO 算法的平均值精度依然保持领先的优势。因此，从总体上看 HAO 算法有着显著的寻优性能，且算法的稳定性同样具有明显优势，HAO 算法的改进是有效的。

表 5　CEC2019 测试函数

Tab. 5　CEC2019 test function

| 函数 | 维度 | 范围 | 理论最优 |
|---|---|---|---|
| CEC01 | 9 | [-8192,8192] | 1 |
| CEC02 | 16 | [-16384,16384] | 1 |
| CEC03 | 18 | [-4,4] | 1 |
| CEC04 | 10 | [-100,100] | 1 |
| CEC05 | 10 | [-100,100] | 1 |
| CEC06 | 10 | [-100,100] | 1 |
| CEC07 | 10 | [-100,100] | 1 |
| CEC08 | 10 | [-100,100] | 1 |
| CEC09 | 10 | [-100,100] | 1 |
| CEC10 | 10 | [-100,100] | 1 |

表 6　CEC2019 测试数据

Tab. 6　CEC2019 test data

| 函数 | 指标 | HAO | AO | SPBO | MFCSA |
|---|---|---|---|---|---|
| CEC01 | 平均值 | 4.35E+04 | 5.50E+04 | 6.64E+04 | 2.39E+11 |
| | 标准差 | 2.41E+03 | 5.77E+03 | 4.19E+04 | 1.80E+11 |
| CEC02 | 平均值 | 1.73E+01 | 1.74E+01 | 1.75E+01 | 1.74E+01 |
| | 标准差 | 3.14E-08 | 1.20E-02 | 1.32E-01 | 6.56E-03 |
| CEC03 | 平均值 | 1.27E+01 | 1.27E+01 | 1.27E+01 | 1.27E+01 |
| | 标准差 | 1.18E-09 | 6.04E-06 | 2.76E-04 | 4.80E-07 |
| CEC04 | 平均值 | 1.83E+01 | 8.30E+02 | 2.50E+03 | 8.22E+01 |
| | 标准差 | 8.11E+00 | 5.94E+02 | 1.68E+03 | 7.10E+01 |
| CEC05 | 平均值 | 1.09E+00 | 1.67E+00 | 2.49E+00 | 1.32E+00 |
| | 标准差 | 5.33E-02 | 3.04E-01 | 6.18E-01 | 9.41E-02 |
| CEC06 | 平均值 | 3.84E+00 | 1.05E+01 | 1.08E+01 | 5.34E+00 |
| | 标准差 | 8.63E-01 | 1.21E+00 | 1.49E+00 | 3.37E-01 |
| CEC07 | 平均值 | 2.46E+02 | 3.25E+02 | 3.85E+02 | 3.29E+02 |
| | 标准差 | 4.45E+01 | 4.13E+01 | 1.54E+02 | 4.20E+01 |
| CEC08 | 平均值 | 4.55E+00 | 5.47E+00 | 5.71E+00 | 4.62E+00 |
| | 标准差 | 3.95E-01 | 6.40E-01 | 6.02E-01 | 4.75E-01 |
| CEC09 | 平均值 | 2.80E+00 | 4.79E+00 | 8.24E+01 | 3.06E+00 |
| | 标准差 | 1.50E-01 | 8.11E-01 | 9.80E+01 | 3.02E-01 |
| CEC10 | 平均值 | 1.93E+01 | 1.98E+01 | 2.02E+01 | 1.97E+01 |
| | 标准差 | 3.65E+00 | 4.71E+00 | 1.20E-01 | 9.47E-01 |

## 4 工程应用

### 4.1 PID 参数优化

为验证 HAO 的实用性，本文通过对在工业控制中广泛使用的 PID 控制器进行参数优化，以验证本文算法的实用性。PID 控制器通过调节比例单元 $K_p$、积分单元 $K_i$、微分单元 $K_d$ 组成的控制量对被控制对象进行调节，其目的是使适应度信息最小，以达到最佳的控制效果。其中，通过 HAO 算法对 PID 控制器优化原理图如图 6 所示。

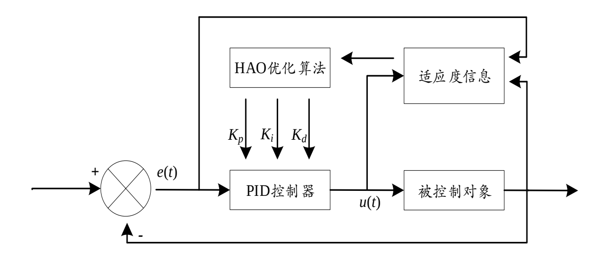



图 6　PID 控制器优化原理图

Fig. 6　PID controller optimization schematic

在 PID 参数优化测试中适应度信息数学模型如式(21)所示。

$$F = \int_0^{\infty}(\omega_1 |e(t)| + \omega_2 u^2(t))dt \tag{21}$$

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d \frac{de(t)}{t}] \tag{22}$$

$$e(t) = in(t) - out(t) \tag{23}$$

其中：$\omega_1$ 和 $\omega_2$ 为[0,1]之间的权值常数，$T_i$ 为积分时间常数，$T_d$ 为微分时间常数，$K_i$=$K_p$/$T_i$，$K_d$=$K_p$×$T_d$。$e(t)$为给定的输入量 $in(t)$和输出量 $out(t)$之间的差值，此外，为了防止出现超调，采用一定的惩罚控制，当 $e(t)$<0 时，适应度信息的数学模型如式(24)所示。

$$F = \int_0^{\infty}(\omega_1 |e(t)| + \omega_2 u^2(t) + \omega_3 |e(t)|)dt \tag{24}$$

其中：$\omega_3$ 为权值，$\omega_3$>>$\omega_1$,通常 $\omega_1$＝0.999，$\omega_2$＝0.001，$\omega_3$=100。在本文中选取二阶延迟系统为被控制对象，其数学模型如式

(25)所示。

$$out(s)=\frac{800}{s^2+1.5s+1.6}e^{-0.1s} \tag{25}$$

在 PID 优化测试中，本文算法(HAO)和 AO、SOA、AOA、HHO、IAO、AOAAO 算法分别对 PID 控制器进行优化，以验证本文算法的实用效果。为了公平性，所有算法迭代次数均为 100，其中 $K_p$、$K_i$、$K_d$ 的取值范围均为[0,300]。其中图 7 为各算法对 PID 控制器进行优化的适应度信息收敛曲线图，表 7 为 PID 优化性能参数。

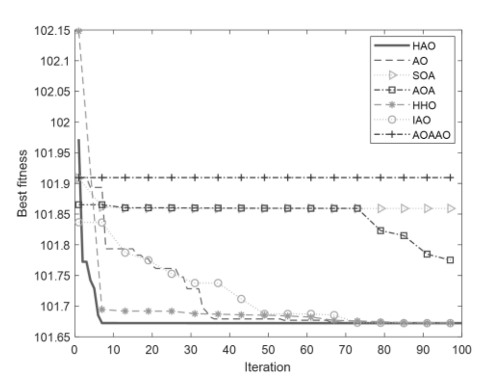



图 7 适应度信息收敛曲线图

Fig. 7 Fitness information convergence curve

通过表 7 的数据可知，HAO 的适应度信息为 101.6723，其适应度值的信息是最小的，因此 HAO 的优化效果较好，虽然 HHO 算法和 IAO 算法对 PID 控制器的优化适应度信息与 HAO 一致，但通过收敛曲线对比图 7 可以看出，HAO 的收敛速度明显优于其他对比算法，HAO 在 10 代以内适应度信息达到最优。因此，相较于其他对比算法，HAO 的实用性更强。

表 7 PID 优化性能参数

Tab. 7 Pid optimized performance parameters

| 算法 | $K_p$ | $K_i$ | $K_d$ | 适应度 |
|---|---|---|---|---|
| HAO | 4.0637 | 0 | 19.1923 | 101.6723 |
| AO | 4.0795 | 0 | 19.1322 | 101.6724 |
| SOA | 0 | 300 | 159.4816 | 101.8592 |
| AOA | 4.1502 | 26.9908 | 54.1113 | 101.7362 |
| HHO | 4.0625 | 0.0007 | 19.187 | 101.6723 |
| IAO | 4.0411 | 0 | 19.257 | 101.6723 |
| AOAAO | 4.2842 | 62.2474 | 71.2376 | 101.7439 |

## 5 结束语

针对 AO 的不足，本文提出一种采用混合搜索策略的阿奎拉优化算法(HAO)。首先利用动态调整切换概率来平衡全局探索与局部开发；其次，利用混沌自适应权重增强算法的全局搜索效率，提升算法的收敛速度；最后，采用改进型差分变异策略，利用适应度值较优个体引领群体中其他个体开展搜索活动，保持了种群的多样性，增强了算法跳出局部最优能力。通过数值实验测试，验证了本文算法在优化精度、收敛速度方面得到了明显地提升，且跳出局部最优的能力得到了增强。最后，通过 PID 参数优化实例测试，验证了本文算法具有较好的实用性。在后续研究中,考虑将本文算法应用于特征选择、无线传感器网络节点定位和图像分割等应用中。

## 参考文献：


[1] Salomon R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms [J]. BioSystems, 1996, 39 (3): 263-278.

[2] Price K V. Differential evolution: a fast and simple numerical optimizer [C]// Proceedings of North American fuzzy information processing. IEEE, 1996: 524-527.

[3] Kennedy J, Eberhart R. Particle swarm optimization [C]// Proceedings of ICNN'95-international conference on neural networks. IEEE, 1995, 4: 1942-1948.

[4] Mirjalili S, Lewis A. The whale optimization algorithm [J]. Advances in engineering software, 2016, 95: 51-67.

[5] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: a gravitational search algorithm [J]. Information sciences, 2009, 179 (13): 2232-2248.

[6] Mirzazadeh M, Eslami M, Zerrad E, *et al.* Optical solitons in nonlinear directional couplers by sine–cosine function method and Bernoulli's equation approach [J]. Nonlinear Dynamics, 2015, 81 (4): 1933-1949.

[7] Singh M, Panigrahi B K, Abhyankar A R. Optimal coordination of directional over-current relays using Teaching Learning-Based Optimization (TLBO) algorithm [J]. International Journal of Electrical Power & Energy Systems, 2013, 50: 33-41.

[8] Pan X, Jiao L. Social cooperation based multi-agent evolutionary algorithm [J]. Journal of Xidian University, 2009, 36 (2): 247-280.

[9] 胡章芳，冯淳一，罗元. 改进粒子群优化算法的移动机器人路径规划 [J]. 计算机应用研究，2021, 38 (10): 3089-3092. (Hu Zhangfang, Feng Chunyi, Luo Yuan. Path planning for mobile robot based on improved particle swarm optimization algorithm [J]. Application Research of Computers, 2021, 38 (10): 3089-3092.)

[10] 宋杰，许冰，杨淼中. 基于自适应步长果蝇优化算法图像分割 [J]. 计算机工程与应用，2020, 56 (04): 184-190. (Song Jie, Xu Bing, Yang Miaozhong. Image segmentation based on adaptive step Drosophila optimization algorithm [J]. Computer Engineering and Application, 2020, 56 (04): 184-190.)

[11] Islam M A, Gajpal Y, ElMekkawy T Y. Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem [J]. Applied Soft Computing, 2021, 110: 107655.

[12] Abualigah L, Yousri D, Abd Elaziz M, *et al.* Aquila optimizer: a novel meta-heuristic optimization algorithm [J]. Computers & Industrial Engineering, 2021, 157: 107250.

[13] Wang S, Jia H, Abualigah L, *et al.* An improved hybrid aquila optimizer and harris hawks algorithm for solving industrial engineering optimization problems [J]. Processes, 2021, 9 (9): 1551.

[14] Zhang Y, Yan Y, Zhao J, *et al.* Chaotic map enabled algorithm hybridizing Hunger Games Search algorithm with Aquila Optimizer [C]// ICMLCA 2021; 2nd International Conference on Machine Learning and Computer Application. VDE, 2021: 1-5.

[15] Ma L, Li J, Zhao Y. Population Forecast of China's Rural Community Based on CFANGBM and Improved Aquila Optimizer Algorithm [J]. Fractal and Fractional, 2021, 5 (4): 190.

[16] Zhang Y J, Yan Y X, Zhao J, *et al.* AOAAO: The Hybrid Algorithm of Arithmetic Optimization Algorithm With Aquila Optimizer [J]. IEEE Access, 2022, 10: 10907-10933.

[17] Dhiman G, Kumar V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems [J]. Knowledge-Based Systems, 2019, 165: 169-196.

[18] Agushaka J O, Ezugwu A E. Advanced arithmetic optimization algorithm for solving mechanical engineering design problems [J]. Plos one, 2021, 16 (8): e0255703.

[19] Aaha B, Sm C, Hf D, *et al.* Harris hawks optimization: Algorithm and applications [J]. Future Generation Computer Systems, 2019, 97: 849-872.

[20] 张伟，王勇，张宁. 采用混合策略的改进学生心理优化算法 [J/OL].

计算机应用研究: 1-8 [2022-04-27]. DOI: 10. 19734/j. issn. 1001-3695. 2021. 1 1. 0630. (Zhang Wei, Wang Yong, Zhang Ning. Improved student psychological optimization algorithm using hybrid strategy [J/OL]. Application Research of Computers: 1-8 [2022-04-27] DOI: 10. 19734/j. issn. 1001-3695. 2021. 1 1. 0630.)

[21] 冯爱武，王勇，付小朋. 采用多模式飞行的乌鸦搜索算法 [J/OL]. 计算机应用研究: 1-9 [2022-04-27]. DOI: 10. 19734/j. issn. 1001-3695. 2021. 12. 0607. (Feng Aiwu, Wang Yong, Fu Xiaopeng Crow search algorithm using multi-mode flight [J/OL]. Application Research of Computers: 1-9 [2022-04-27] DOI: 10. 19734/j. issn. 1001-3695. 2021. 12. 0607.)